A Novel LDPC Decoder for DVB-S2 IP

Stefan Müller, Manuel Schreger, Marten Kabutz
THOMSON
- System Architecture Group -
Herman-Schwer-Str. 3
78048 Villingen-Schwenningen, Germany
Email: {Stefan.Mueller, Manuel.Schreger, Marten.Kabutz}@thomson.net

Matthias Alles, Frank Kienle, Norbert Wehn
University of Kaiserslautern
- Microelectronic Systems Design Research Group -
Erwin-Schroedinger-Str.
67663 Kaiserslautern, Germany
Email: {alles, kienle, wehn}@eit.uni-kl.de

***Abstract*—In this paper a programmable Forward Error Correction (FEC) IP for a DVB-S2 receiver is presented. It is composed of a Low-Density Parity Check (LDPC), a Bose-Chaudhuri-Hoquenghem (BCH) decoder, and pre- and postprocessing units. Special emphasis is put on LDPC decoding, since it accounts for the most complexity of the IP core by far.**

We propose a highly efficient LDPC decoder which applies Gauss-Seidel decoding. In contrast to previous publications, we show in detail how to solve the well known problem of superpositions of permutation matrices. The enhanced convergence speed of Gauss-Seidel decoding is used to reduce area and power consumption. Furthermore, we propose a modified version of the λ-Min algorithm which allows to further decrease the memory requirements of the decoder by compressing the extrinsic information.

Compared to the latest published DVB-S2 LDPC decoders, we could reduce the clock frequency by 40% and the memory consumption by 16%, yielding large energy and area savings while offering the same throughput.

***Index Terms*—Forward Error Correction, Soft Decision Decoding, LDPC, DVB-S2, Check Node approximation.**

I. INTRODUCTION

The DVB-S2 specification [1], [2] is the world's first standard using LDPC codes for the FEC. Together with an outer BCH decoder, it allows for outstanding communications performance. Because of the long codewords of 64 800 bits, the implementation complexity of the LDPC decoder is very challenging. It accounts by far for the most complexity in a DVB-S2 FEC. Therefore, a highly efficient LDPC decoder implementation is mandatory.

LDPC decoding is an iterative process. Two sets of computations have to be performed per iteration: check node updates and bit/symbol node updates. Decoding architectures based on the Gallager algorithm [3] execute these computations in two distinct steps or phases (so called Two Phase Message Passing algorithm). By means of the Gauss-Seidel algorithm, which is also known as "staggered decoding", "Turbo Decoding Message Passing" (TDMP), "shuffled decoding" or "layered decoding" [4]–[6], intermediate results are used within the same iteration for the same computation type. This technique is applicable to partially parallel decoder architectures, where not all computations of one node type are executed in parallel. Gauss-Seidel decoding improves the convergence behavior and it is possible to reduce the number of required iterations by up to 50% at the same bit error rate (BER) performance. Due to the increased efficiency the latency, the parallelism of the decoder or the clock frequency can be reduced, saving both power and area. For highly efficient decoder implementations it is furthermore necessary to use suboptimal check node approximations of low complexity, e.g., the λ-Min algorithm [7].

Unfortunately, there is a special case in the parity check matrix of DVB-S2 that raises difficulties when applying the Gauss-Seidel algorithm in partially parallel decoder architectures. DVB-S2 LDPC codes are based on submatrices (permuted identity matrices) and superpositions of these submatrices (called superposed submatrices hereafter), cf. Section III. The Gauss-Seidel algorithm cannot be applied for these superposed submatrices. DVB-S2 LDPC architecture publications applying the Gauss-Seidel algorithm [8]–[13] do not address this problem.

In this paper, we present a new LDPC decoder that applies a novel method for Gauss-Seidel decoding of the DVB-S2 codes. A two phase message passing is used only for the superposed submatrices. For all other submatrices, Gauss-Seidel decoding is applied resulting to increase decoder efficiency. The superposed submatrix problem is solved without any additional processing delay, see Section IV. Furthermore, we propose a modified version of the λ-Min algorithm [7] that further reduces the size of the extrinsic memory compared to the latest published DVB-S2 decoders, cf. Section VI. Bit accurate simulations show the communications performance of our approach.

The presented IP comprises all needed pre- and postprocessing units of a DVB-S2 FEC system, such as log-likelihood ratio (LLR) computation, deinterleaving, LDPC decoding, BCH decoding, physical and baseband descrambling and data preparation for the backend MPEG decoder. The overall area after place and route is 13.1mm^2 in a 90nm CMOS technology. Compared to the latest published DVB-S2 decoder [13] we could reduce the clock frequency by 40% while the memory consumption is reduced by 16%, see Section VII.

II. SYSTEM OVERVIEW

Figure 1 shows an overview of the presented FEC design. Three modulation constellations (QPSK, 8PSK and H8PSK) and all eleven code configurations are supported for the codeword length of 64 800 bits. The FEC is capable of decoding

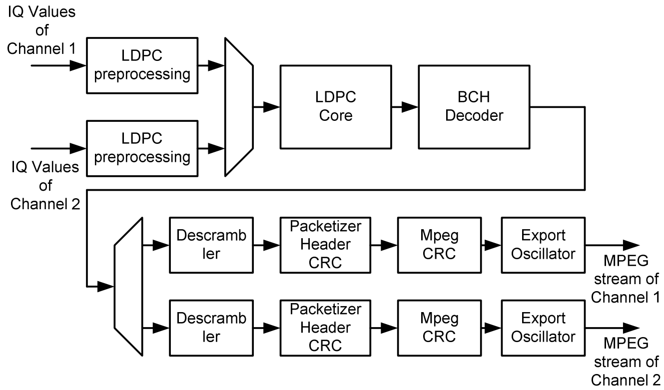

Fig. 1. DVB-S2 FEC Decoding System

two independent DVB-S2 data streams at 30 MBaud each.

The "Inphase" and "Quadrature" (I/Q) signals are received by the LDPC preprocessing unit, which computes the LLRs, deinterleaves the data and saves the LLRs in internal memory. The saved frames are iteratively decoded by the LDPC core unit and transmitted to the BCH decoder. The postprocessing blocks perform descrambling, packetizing and a cyclic redundancy check (CRC). Because the decoded data is processed in a burst wise manner, the packets must be smoothed in time so that they are transmitted at regular intervals. This then corresponds to the timing of the packets as they were received at the transmitter. This is performed by the Export Oscillator in Figure 1 using a FIFO of size 16 kBytes along with a numerical controlled oscillator. The output data stream is then passed to the MPEG decoder.

III. DVB-S2 LDPC Codes

An LDPC code is defined by a sparse parity check matrix H of size $M \times N$. It can be represented by a bipartite graph, the so called Tanner graph [14]. The Tanner graph consists of two types of nodes, variable and check nodes. The check node j is connected to variable node i whenever element h_{ij} in H is 1, with $j \in \{0, \ldots, M-1\}$ and $i \in \{0, \ldots, N-1\}$. No connection exists when h_{ij} is 0.

The parity check matrices in DVB-S2 are architecture-aware, which means that the codes are suitable for partially parallel decoder architectures with a parallelism of up to $S = 360$. Therefore the parity check matrix is divided into submatrices of size $S \times S$, which means that the variable nodes and check nodes are divided into groups of size S. The submatrix itself defines the permutation π between the node groups. Figure 2 shows an example of an architecture-aware irregular repeat accumulate code as used in the DVB-S2 standard. The bits in the codeword, represented by variable nodes, are separated into information nodes and parity nodes. Furthermore S information nodes u_i and S parity nodes y_i are grouped together, forming the information node group ING_i and parity node group PNG_i, respectively. The parity nodes are of degree two for all code rates R. The information nodes consist of two subsets, one with degree of $w_c(R)$ and one with degree three. The check equations are represented by check nodes and grouped together as well. They all are of degree

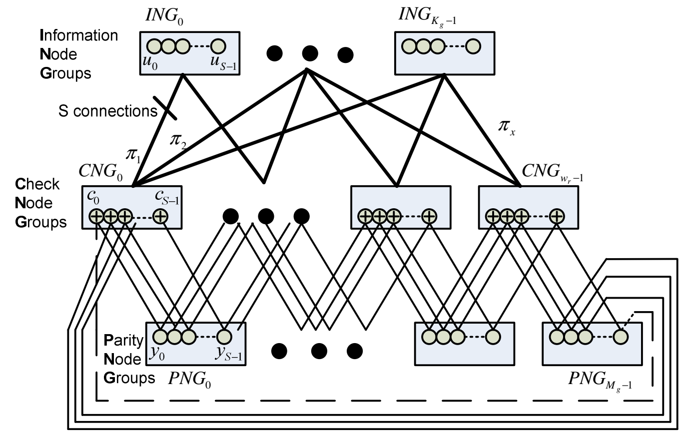

Fig. 2. Architecture-aware irregular repeat accumulate code used in DVB-S2

$w_r(R)$. The check node groups are connected to information node groups and parity node groups via permutations π, where the permutation varies between the groups. The connection between information node groups and the check node groups are semi-random defined according to the DVB-S2 encoding rules [1], [2]. The connection of the parity node group and the check node group is a fixed zigzag pattern, which is the result of the accumulator encoding procedure [15]. The permutation in DVB-S2 can be represented by a $S \times S$ cyclic shift of the identity matrix I denoted as I^x, where x is the number of left cyclic shifts. For all DVB-S2 LDPC codes there are three types of submatrices of size $S \times S$ in their parity check matrices:

- The submatrix 0
- Submatrices I^x with $x \in \{0, \ldots, S-1\}$
- Submatrices composed of superpositions, e.g. $I^x + I^y$ with $x \neq y$, and $x, y \in \{0, \ldots, S-1\}$.

$H'(m, n)$ denotes the submatrix of the parity check matrix corresponding to check node group $m \in \{0, \ldots, \frac{M}{S} - 1\}$ and variable node group $n \in \{0, \ldots, \frac{N}{S} - 1\}$. So, the parity check matrix can be written as

$$H' = \left[\underbrace{\begin{matrix} I^0 & I^1 & \ldots & 0 \\ I^2 & I^3 + I^4 & \ldots & I^5 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & I^9 & \ldots & I^2 \\ I^7 & 0 & \ldots & I^1 \end{matrix}}_{A} \middle| \underbrace{\begin{matrix} I^0 & 0 & \ldots & 0 & I^{1^*} \\ I^0 & I^0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & I^0 & 0 \\ 0 & 0 & \ldots & I^0 & I^0 \end{matrix}}_{T} \right] \quad (1)$$

Matrix part A describes the semi-random part of information node groups and matrix part T the zigzag pattern or stair-case part of the parity node groups. The matrix entry I^{1^*} denotes the dotted connection between the last parity node group and the first check node group of Figure 2, that must not to be processed.

IV. Modified Gauss-Seidel Decoding

The Gauss-Seidel method is a technique used to solve a linear system of equations in an efficient way. The fundamental principle is to use intermediate results directly in the next computation in order to accelerate the convergence behaviour. The increased convergence speed allows a reduction in the number of LDPC iterations by up to 50%, giving a much higher decoder efficiency.

One problem of applying Gauss-Seidel to the DVB-S2 code is the handling of superposed submatrices. In a partially parallel decoder architecture with a parallelism degree of S, the S check nodes of each check node group in H' are computed in parallel. Each check node group uses the results of previously computed groups. If the check node group contains superposed submatrices it is not possible to apply the Gauss-Seidel principle inside that submatrix, e.g., in $H'(1,1)$ of equation (1) it is not possible to use the intermediate results of I^3 in I^4 and vice versa. The updated results are not yet available when actually needed.

The architecture presented in [8] avoids the problem by applying the Gauss-Seidel technique only on the staircase part T of the parity check matrix H. The architecture in [10] applies Gauss-Seidel to all variable nodes, but does not show the treatment of superposed submatrices. In the following, we show how to solve the problem of superposed submatrices without any latency penalty.

For each variable node i, an accumulator A_i is introduced. It is initialized with the received channel reliabilities L^i_{ch}, expressed in LLRs,

$$A^i = L^i_{\text{ch}}. \tag{2}$$

For the variable node group n, we define the accumulator group A^n_{VNG} that contains the accumulators of all variable nodes contained in the variable node group n. During decoding, the accumulators always contain the a-posteriori value. As mentioned before, partially parallel decoder architectures only work on the node groups.

For each decoding iteration, $\frac{M}{S}$ decoding sub-iterations have to be performed, corresponding to the number of check node groups. For each check node group m, new extrinsic reliabilities are computed using the a-posteriori value saved in the accumulator groups and the extrinsic information. Let $C(m)$ be the set of permuted identity matrices that are contributing to the check node group m. Furthermore, let $f(k)$ give the corresponding variable node group n for the permuted identity matrix $k \in C(m)$. New a-posteriori values and extrinsic information are computed as follows:

- For each permuted identity matrix $k \in C(m)$ read the corresponding accumulator values $A^{f(k)}_{\text{VNG}}$ and perform the permutation:

$$A^{k*}_{\text{VNG}} = \pi\left(A^{f(k)}_{\text{VNG}}\right) \tag{3}$$

- For each $k \in C(m)$ compute the intrinsic reliability by subtracting the extrinsic information group L^k_{extr} from the temporary saved accumulator value. For the first iteration L^k_{extr} is zero for all k.

$$L^k_{\text{int}} = A^{k*}_{\text{VNG}} - L^k_{\text{extr}} \tag{4}$$

- For each $k \in C(m)$ compute new extrinsic reliability information:

$$L^{k*}_{\text{extr}} = 2\tanh^{-1}\left(\prod_{l \in C(m)\setminus k} \tanh\left(\frac{L^l_{\text{int}}}{2}\right)\right) \tag{5}$$

- Compute new a-posteriori information for each $k \in C(m)$:

$$A^{k*}_{\text{VNG}} = A^k_{\text{int}} + L^{k*}_{\text{extr}} \tag{6}$$

- Permute back the a-posteriori information for each $k \in C(m)$:

$$A^{k'}_{\text{VNG}} = \pi^{-1}\left(A^{k*}_{\text{VNG}}\right) \tag{7}$$

- Distinguish now between normal case and superposition:
  - In case variable node group $f(k)$ is unique for all $k \in C(m)$ (no superposed submatrices):

$$A^{f(k)}_{\text{VNG}} = A^{k'}_{\text{VNG}} \tag{8}$$

  - In case variable node group $f(k)$ is used multiple times for $k \in C(m)$ (superposed submatrices), update the partially calculated a-posteriori information $A^{k'}_{\text{VNG}}$ with missing extrinsic differences caused by superposition of multiple permutations $j \in C(m)$:

$$\begin{aligned} A^{f(k)}_{\text{VNG}} = A^{k'}_{\text{VNG}} & \\ + \sum_{\substack{j \mid f(j)=f(k), \\ j \neq k}} & \pi^{-1}\left(L^{j*}_{\text{extr}} - L^j_{\text{extr}}\right) \end{aligned} \tag{9}$$

- Overwrite the old extrinsic reliabilities:

$$L^k_{\text{extr}} = L^{k*}_{\text{extr}} \tag{10}$$

The a-posteriori reliability is available after each sub-iteration and corresponds to the accumulator group values A^k_{VNG}. Due to the fact that the a-posteriori sum is updated during each sub iteration, the decoding process can be stopped at any time. The hard decisions of a variable node group is done by evaluating the sign of A^k_{VNG}.

V. LDPC Decoder Architecture

Figure 3 shows the decoder architecture based on the algorithm described in Section IV. The accumulator memory contains the a-posteriori sum and is initialized with the received channel LLRs according to equation (2). These values are updated at each subiteration. Therefore the accumulator values are loaded to the shifter and shifted according to the permutation matrices. The corresponding extrinsic information is loaded from the edge memory. In case the edge memory is compressed as explained in Section VI, it is first necessary to decompress the RAM content. For the first iteration, the extrinsic information is set to zero. The extrinsic information is subtracted from the shifted a-posteriori value, see equation (4). The resulting intrinsic value is fed to the check node processing unit (CNU) unit and to a multiplexer. If the submatrix is defined over only one permutation matrix, the intrinsic value is fed to the FIFO, i.e., the result of the addition is used. If

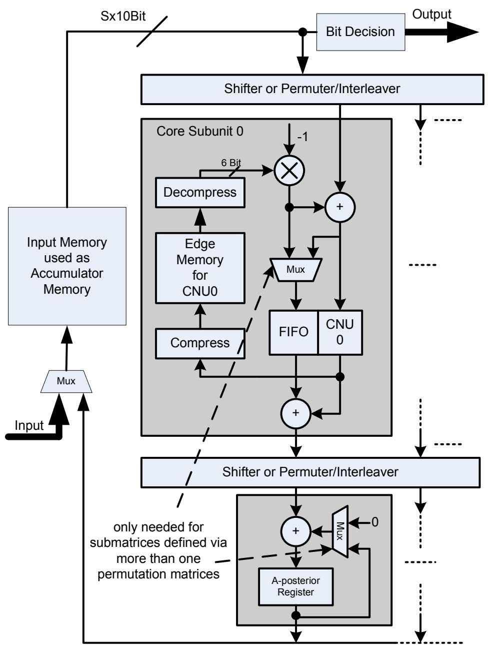

Fig. 3. Hardware architecture for modified Gauss-Seidel decoder

the submatrix is defined via more than one permutation matrix, the first intrinsic value is fed to the FIFO and afterwards the FIFO is fed with the missing negative extrinsic values of other permutations for that submatrix. This prepares the computation inside the brackets of equation (9).

Now the CNU computes new extrinsic values according to equation (5). These values are compressed and saved in the edge memory. At the output of the CNU, the new extrinsic value is added to the output of the FIFO and passed to the shifter, compare equation (6) and (7). The output of the shifter is fed to an adder. Again it is distinguished between one or more permutation matrices in one submatrix. If the submatrix is defined via one permutation matrix, the output value of the shifter is directly stored in the a-posteriori register (no adding), as shown in equation (8). If the submatrix is defined via more than one permutation matrix, the first value is loaded to the a-posteriori register and the following extrinsic differences are added to that value according to equation (9). This requires that multiple permutations in one submatrix are processed in consecutive order. Finally, the old a-posteriori value in the accumulator memory is overwritten with the updated a-posteriori register value.

VI. Extrinsic memory compression

A key unit of the LDPC decoder is the computation of the extrinsic values in the CNUs according to equation (5). Due to the high implementation complexity of this equation, suboptimal algorithms are used, which still yield good communications performance. A good approximation is the λ-Min approximation [7]. Where only the λ smallest incoming magnitudes are used to compute the extrinsic information.

The DVB-S2 LDPC decoder has to store up to $285\,120$ LLRs for the extrinsic information. With a quantization of 6 bits for each LLR, a total memory size of $1\,710\,720$ bits is required for the extrinsic information.

One advantage of the λ-Min algorithm is that only $\lambda+1$ different magnitudes are computed. As proposed in [7], it is hence possible to compress the extrinsic memory. To store the output of the CNU, it is only necessary to store w_r sign bits, $\lambda+1$ magnitudes, and λ indices defining where the first λ magnitudes are to be found in the serial output stream of the CNU. The $(\lambda+1)$th magnitude is used on all other positions in the output stream.

The memory reduction depends on the row weight w_r and on the bit width of the LLRs. Let us assume that the bit width of one LLR value is 6 bit and $w_r = 30$; the memory size of one uncompressed LLR vector is 180 bits, the compressed vector for 3-Min decoding requires only 65 bits (30 sign bits, 4*5 magnitude bits, 3*5 index bits).

For low code rates however, w_r can be as low as four. Using the compressed storage method with the 3-Min algorithm as proposed in [7] would result in an increased memory consumption. In order to achieve a maximum reduction of memory, we therefore modified the 3-Min algorithm to obtain less different magnitudes.

Let the input to the check node processing unit be L^i_{int} and the extrinsic output be L^i_{extr}, $i = 0,1,...,w_r(R)-1$. The three smallest values in the set of $|L^i_{\text{int}}|$ are selected and let their corresponding indices be j_0, j_1, j_2, where $|L^{j_0}_{\text{int}}| < |L^{j_1}_{\text{int}}| < |L^{j_2}_{\text{int}}|$. Furthermore, let $s_i = \operatorname{sign}(L^i_{\text{int}})$ and $S = \prod_{i=0}^{w_r(R)-1} s_i$. Four magnitudes are to be calculated for the 3-Min algorithm but we only calculate three different extrinsic output values L'^i_{extr}. The fourth value, conventionally obtained by combination of all three minimas, is approximated by $|L'^{j_2}_{\text{extr}}|$, therefore the extrinsic output is calculated by

$$L^i_{\text{extr}} = s_i \times S \times \begin{cases} |L'^{j_0}_{\text{extr}}| & \text{for } i = j_0 \\ |L'^{j_1}_{\text{extr}}| & \text{for } i = j_1 \\ |L'^{j_2}_{\text{extr}}| & \text{for } i \neq j_0, j_1 \end{cases} . \tag{11}$$

To exploit the compression logic only two magnitudes are selected for low code rates ($R < 1/2$),

$$L^i_{\text{extr}} = s_i \times S \times \begin{cases} |L'^{j_0}_{\text{extr}}| & \text{for } i = j_0 \\ |L'^{j_1}_{\text{intr}}| & \text{for } i \neq j_0 \end{cases} . \tag{12}$$

Without any compression $1\,710\,720$ bits (100%) are required for the extrinsic LLRs with a 6 bit quantization. When computing three magnitudes for all code rates only $1\,123\,200$ bits (66%) would be required. By employing our new mixed approximation approach the memory requirement can be even further reduced to $972\,000$ bits (53%).

VII. Results

A. Simulation Results

This section shows simulation results of the forward error correction decoder according to Figure 1. Figure 4 and Figure 5 show the bit accurate BER performance vs. the signal-to-noise ratio (SNR) for the two supported modulation schemes

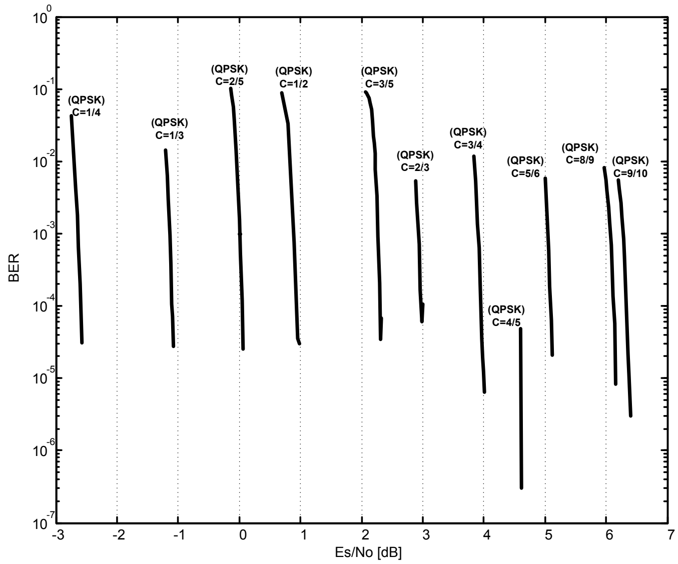

Fig. 4. BER simulation results of all code rates using QPSK modulation

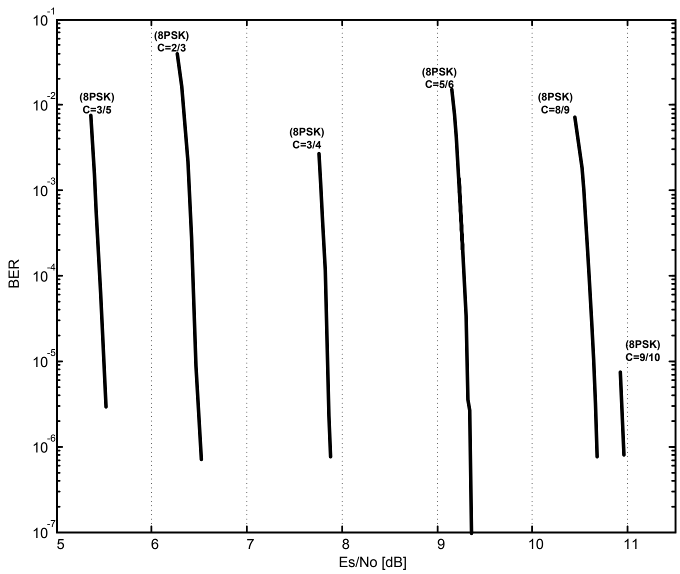

Fig. 5. BER simulation results of all code rates using 8PSK modulation

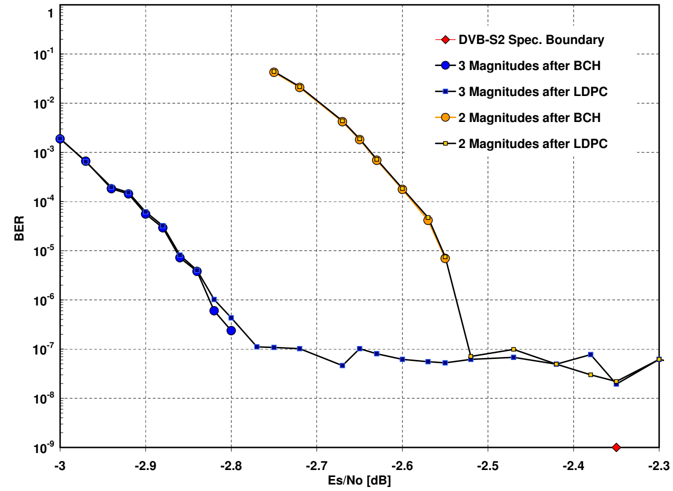

Fig. 6. BER simulation results for $R = 1/4$ using QPSK modulation

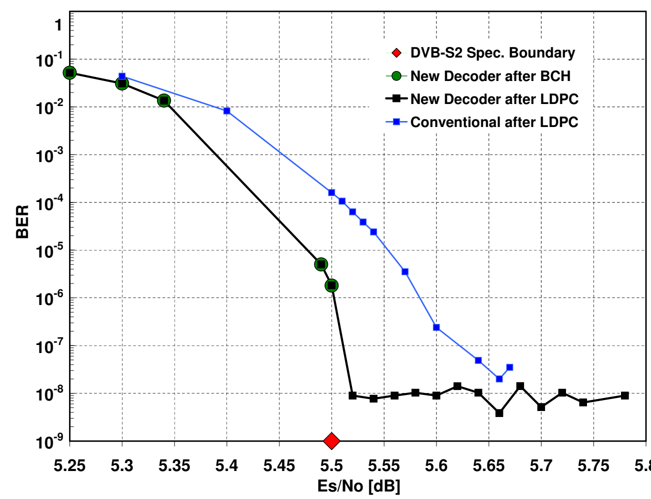

Fig. 7. BER simulation of $R = 3/5$ with 8PSK modulation

(QPSK, 8PSK) at a constant block size of $N = 64\,800$. In our simulation model, we add white Gaussian Noise (AWGN) to the I/Q signals. These signals are sampled at ideal time and quantized to 10 bit. The sampled values are then passed to the forward error correction, where they are mapped to 6 bit LLRs and decoded as presented in Section IV. Our investigations showed, that 6 bit is the minimum required bit quantization in order to achieve the required bit error performance.

Figure 6 shows the bit error rate for $R = 1/2$ and QPSK modulation using our modified calculation scheme (at 50 iterations). The performance loss is about 0.3 dB when computing two magnitudes instead of three. The error floor is eliminated by the BCH decoder at -2.55 dB still satisfying the DVB-S2 specification. The code rate $R = 3/5$ in 8PSK mode is the most time-critical configuration in the presented system, since this code has the largest number of edges in the Tanner graph. Figure 7 depicts the BER performance with and without BCH decoding for that code configuration. The maximum number of iterations was limited to 40. One can notice a significant error floor of 10^{-8} starting at 5.5 dB without BCH decoder. The error floor is eliminated by the BCH decoder at 5.5 dB, which is exactly the specification requirement of DVB-S2. In comparison to a conventional decoding architecture with two phase decoding the coding gain is improved around 0.1 dB, see Figure 7. To satisfy the DVB-S2 specification with two phase decoding, more iterations are required, resulting either in a lower throughput at a given clock frequency or a higher energy consumption for fixed throughputs since the clock frequency would have to be increased.

B. Implementation Results

The forward error correction design was implemented in TSMC 90nm CMOS technology. Implementation results of the proposed decoder in 90nm technology [12], and the latest published DVB-S2 decoder IP [13] (65nm technology) are summarized in Table I. The presented FEC IP core has a total area of 13.1mm^2 after place and route. Even with compression of the extrinsic LLRs, 67% of the core area are required for the

TABLE I
IMPLEMENTATION RESULTS

Paper	This	[12]	[13]	This†1	This†2
Technology [nm]	90	90	65	65	-
No. of streams	2	1	1	2	1
Parallelism	90	360	180	90	180
Air Throughput [Mbit/s]	2*90	135	135	2*90	135
Frequency [MHz]	270	300	**174**	270	**105**
Memory capacity [Mbits]	3.59	2.83	**3.18**	3.60	2.80/**2.68**†3
Memories No. cuts	37	65	91	50	-
Memory area [mm^2]	8.78	5.20	3.00	4.34	-
Gate Count [kGates]	650	-	-	668	-
Power [mW]	477/853	980	476	-	-
Total Area [mm^2]	13.1	15.8	**6.07**	**6.03**†4	-

†1 for comparison: design shrunk to 65nm, synthesis only
†2 for comparison: same parameters as [13]
†3 excluding export oscillator
†4 estimated after place and route

memories. The main reason is the large block size of 64 800 used in DVB-S2. Compared to the 90nm decoder of [12] we require less power and area while offering a higher throughput. Power numbers are obtained assuming a toggling rate for all flip-flops of 25% under best case (853mW) and worst case timing conditions (477mW). We furthermore synthesized the decoder in a 65nm process. In this case the overall area of our core is reduced to 6.03mm^2. Compared to the decoder of [13] we could reduce the decoder parallelism, what allows for a simplified routing of the barrel shifters.

Comparing the different approaches is difficult though. Different number of channels, parallelisms, and other requirements lead to different decoder implementations. Therefore, we adapted the presented IP to one channel with an LDPC decoder parallelism of 180, which is the configuration of [13]. The obtained numbers show the efficiency of our approach. Compared to [13] the clock frequency could be reduced by 40% from 174 MHz to 105 MHz to obtain the same throughput. Furthermore, when excluding the export oscillator, the overall memory consumption is reduced by 16% from 3.18 Mbits to 2.68 Mbits. The clock frequency of the presented receiver IP is determined by the code rate $3/5$ and 8PSK modulation, since decoding this code takes the most clock cycles to meet the DVB-S2 specification.

The layout of the overall IP core is depicted in Figure 8. It contains all components as shown in Figure 1, such as LDPC decoder, BCH decoder, descrambler, CRC, etc. As mentioned before, the biggest part of design are the memories. They are placed around the logic of LDPC core and the pre- and postprocessing blocks.

VIII. CONCLUSION

In this paper we presented a novel LDPC decoder, which was used within an FEC IP core for DVB-S2. The decoder is able to apply Gauss-Seidel decoding in an efficient way even when the parity check matrix contains superpositions of permutation matrices. Furthermore, the λ-Min algorithm was modified to allow a higher compression rate of the extrinsic memory. The proposed techniques allow a reduction in power and area consumption compared to previous publications.

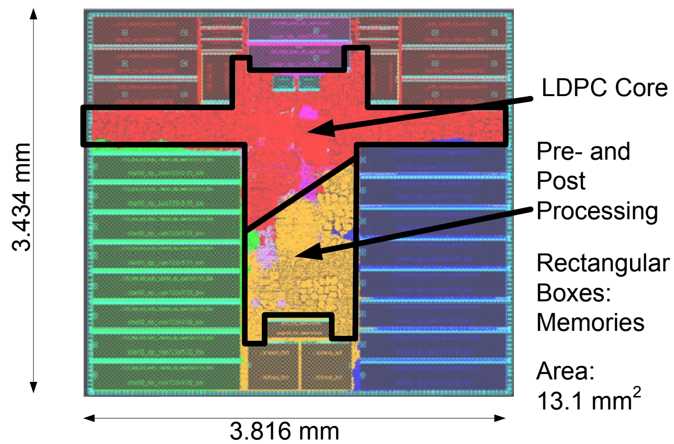

Fig. 8. Layout of forward error correction design

REFERENCES

[1] European Telecommunications Standards Institute (ETSI), "Digital Video Broadcasting (DVB) Second generation framing structure for broadband satellite applications; EN 302 307 V1.1.1," www.dvb.org.

[2] A. Morello and V. Mignone, "DVB-S2: The Second Generation Standard for Satellite Broad-Band Services," *Proceedings of the IEEE*, vol. 94, pp. 210–227, 2006.

[3] R. Gallager, "Low-density parity-check codes," *Information Theory, IRE Transactions on*, vol. 8, pp. 21–28, 1962.

[4] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam, "High Throughput Low-Density Parity-Check Decoder Architectures," in *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, vol. 5, 2001, pp. 3019–3024 vol.5.

[5] M. Mansour and N. Shanbhag, "High-Throughput LDPC Decoders," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 11, pp. 976–996, 2003.

[6] D. Hocevar, "A Reduced Complexity Decoder Architecture via Layered Decoding of LDPC Codes," in *Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on*, 2004, pp. 107–112.

[7] F. Guilloud, E. Boutillon, and J. Danger, "λ-Min Decoding Algorithm of Regular and Irregular LDPC Codes," in *Proc. 3nd International Symposium on Turbo Codes & Related Topics*, Brest, France, Sep. 2003, pp. 451–454.

[8] F. Kienle, T. Brack, and N. Wehn, "A synthesizable IP Core for DVB-S2 LDPC Code Decoding," in *Design, Automation and Test in Europe. Proceedings*, 2005, pp. 100–105 Vol. 3.

[9] A. Segard, F. Verdier, D. Declercq, and P. Urard, "A DVB-S2 compliant LDPC decoder integrating the Horizontal Shuffle Scheduling," in *Intelligent Signal Processing and Communications, 2006. ISPACS '06. International Symposium on*, 2006, pp. 1013–1016.

[10] J. Dielissen, A. Hekstra, and V. Berg, "Low cost LDPC decoder for DVB-S2," in *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, vol. 2, 2006, pp. 1–6.

[11] M. Gomes, G. Falcao, V. Silva, V. Ferreira, A. Sengo, and M. Falcao, "Flexible Parallel Architecture for DVB-S2 LDPC Decoders," in *Global Telecommunications Conference. GLOBECOM '07. IEEE*, 2007, pp. 3265–3269.

[12] P. Urard, E. Yeo, L. Paumier, P. Georgelin, T. Michel, V. Lebars, E. Lantreibecq, and B. Gupta, "A 135Mb/s DVB-S2 Compliant Codec Based on 64800b LDPC and BCH Codes," in *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, 2005, pp. 446–609 Vol. 1.

[13] P. Urard, L. Paumier, V. Heinrich, N. Raina, and N. Chawla, " A 360mW 105Mb/s DVB-S2 Compliant Codec based on 64800b LDPC and BCH Codes enabling Satellite- Transmission Portable Devices," in *Solid-State Circuits Conference, 2008. Digest of Technical Papers. ISSCC. 2008 IEEE International*, 2005, pp. 310–311.

[14] R. Tanner, "A Recursive Approach to Low Complexity Codes," *Information Theory, IEEE Transactions on*, vol. 27, pp. 533–547, 1981.

[15] H. Jin, A. Khandekar, and R. McEliece, "Irregular Repeat-Accumulate Codes," *Second International Conference on Turbo Codes, Brest, France*, Sep. 2000.